CONNEXIONS™

The Interoperability Report

July 1989 Volume 3, No. 7

ConneXions— The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

ConneXions is published by Advanced Computing Environments, 480 San Antonio Road, Suite 100, Mountain View, CA 94040, USA. Phone: 415-941-3399.

ISSN 0894-5926

From the Editor

Continuing our series *Components of OSI,* we come to the Transport Layer. The article is by Nancy Hall from the University of Wisconsin. Since the OSI Transport Layer offers similar functionality to the Transmission Control Protocol (TCP) in the Internet Protocol Suite, Nancy contrasts and compares the two protocols.

We have discussed both interoperability and conformance testing in previous editions of *ConneXions*. (See in particular the August 1988 issue). This month, Marshall Rose offers his perspective on testing, and argues that there is a significant difference between conformance testing and interoperability testing. Marshall Rose recently joined NYSERNet as Senior Scientist for the Western Development Office.

The next INTEROP™ conference and exhibition will be held in San Jose, California from October 2 through 6, 1989. An advance program should be arriving in you mailbox any day now. In this issue of *ConneXions* we bring you some highlights of the cooperative demonstrations that will be taking place on the exhibit floor during INTEROP 89.

On pages 16 and 17, we list some recent publications in the field of computer networking. Following that is another in our series of book reviews, although I should point out that in this particular case it serves more as a warning against the book in question. When Charles Spurgeon from the University of Texas at Austin sent me the review he said: "Reading this book was one of the strangest experiences I can remember having with respect to technical literature. It appears that the author never bothered to check his "facts" against the specifications for Ethernet before writing an entire book on the technology. The end result is one of the weirdest books I've ever read. I can't imagine how McGraw Hill published this thing without a technical review, but the evidence insists that that's just what they did."

James VanBokkelen of FTP Software describes some recent workstation-oriented enhancements to the Telnet protocol. In a future issue we will bring you a more in-depth description of Telnet, and it is also one of the topics to be discussed at INTEROP 89. A new IETF working group has recently been formed to further develop Telnet. It is chaired by Dave Borman from Cray Research, and information about the working group can be found on page 22.

Components of OSI: The Transport Layer

by Nancy Hall, University of Wisconsin

Introduction

The purpose of this article is to introduce you to the ISO/CCITT Open Systems Interconnection (OSI) Transport Layer. We describe the purposes of the Transport Layer, the services it provides, and some of the protocol mechanisms by which it provides these services. The OSI Transport protocol and services are contrasted and compared with their counterparts in the DARPA Internet protocol suite, the Transmission Control Protocol (TCP) and the services that it provides. In the interest of space, many issues have not been discussed here, but a bibliography appears at the end of this article.

Jargon

Before we continue, let us take a few paragraphs to describe the general format of the OSI standards and to introduce some of the OSI jargon. For each layer of the OSI architecture, there are two standards: one describes the *services* provided by the layer; the other describes the *protocols* that support the layer services. The software and hardware that implement the protocol and services is called the *provider*. In the standards documents, the services are described in terms of *primitives* and their *parameters*. The primitives and their parameters indicate what information flows between the user and the provider of the service; they say nothing about the method by which this information is conveyed. In other words, the service standards do not standardize programming interfaces.

Four types of primitives are defined, *request*, *indication*, *response* and *confirm*. A *request* is the primitive initiated by a user to request a service from the provider. An *indication* is initiated by the provider to inform the user of some condition, such as the presence of incoming data. Some indications require an action on the part of the user; this is the *response* primitive. A *confirm* primitive is initiated by the provider to convey to the user the information that was given by the peer user to the peer provider in a *response* primitive. The *confirm* primitive implicitly means that the original *request* has been completed.

A packet of user data, or a unit of data from the point of view of the transport service user, is called a *transport service data unit*, or TSDU. The data passed between the user and the provider in one service primitive is a TSDU. Similarly, a packet that is passed to the network service provider in a network service primitive is called an NSDU.

The protocol standards describe the functions required of the *provider* in order to provide the services. Most importantly, it describes the format of the packets exchanged by the providers. These packets are called *protocol data units* or PDUs. A PDU in the transport protocol is called a TPDU.

Services

The Transport Layer is the 4th layer in what has become the canonical representation of a network architecture, the OSI 7-layer model. The 4th layer is responsible for end-to-end transfer of data. This layer provides several services with varying amounts of pith, ranging from guarantees about data arriving at their destination in the order sent, to no guarantees whatsoever about data corruption or loss.

In all cases however, the Transport Layer makes invisible to the higher layers the composition of the layers below it and the topology of the network on which these services are provided. The service provided by this layer is intended to be independent of all qualities of the network service or services on which it runs. It is the job of the Transport Layer to choose and use the proper network service(s) in order to meet charging requirements and requirements about the quality of the service provided. It is also the Transport Layer that detects end-to-end data corruption errors. (The Network Layer may detect the corruption of data as it crosses a link in the network, but corruption that occurs in a switch or gateway may not be detected by the Network Layer.)

Quality Of Service

The quality of service (QOS) is the subject of current work in ISO. The idea is that by means of a set of QOS parameters on service primitives, the user of the transport service tells the provider what quality of service is required, and the provider is bound to do its best to maintain that quality of service. The format, values, and exact semantics of the QOS parameters are not yet standardized, but the standards do present a set of transport QOS parameters and some of the characteristics of a transport connection that these parameters are meant to express. The set includes such things as the probability that a transport connection will fail to be established within a specified time, transit delay, relative priorities of transport connections, and protection (or security). The means by which the quality of service is maintained is not specified. The QOS protection parameter is the topic of new work in ISO, in the context of a security architecture that spans all layers of the OSI model. A detailed discussion of the topics of QOS and security is beyond the scope of this article.

In the next several paragraphs we describe the transport services in detail. The OSI Transport Layer offers a set of seemingly independent useful functions to the user of the Transport Layer. These functions are grouped into two services: the *connectionless-mode* service and the *connection-mode* service. The OSI Transport Layer does not attempt to provide all useful combinations of functions. For example, it does not (yet) provide an acknowledged connectionless-mode service, or a multi-destination data transfer service. New services could and without doubt will be added to all layers of the OSI architecture, for example, work is indeed progressing in ISO in the area of multi-way transmission.

Connectionless-mode Service

The OSI connectionless-mode transport service is a datagram service, or one that transfers data in discrete packets ("datagrams") from a source to a destination without any guarantee of delivery, without any indication to the sender when the packets are delivered to the destination, without any notion of order among the packets, and without any notion of an association or "connection" between the source and destination entities. What it *does* offer is the optional guarantee that each packet that arrives at its destination arrives intact: it is not truncated or corrupted. This guarantee may be selected or rejected at the option of the user of the service. This kind of service comprises two service primitives. In OSI argot, these two primitives are called `T-UNITDATA.request` (a send operation) and `T-UNITDATA.indication` (a receive operation). Each primitive has four parameters: the source transport address, the destination transport address, the quality of service, and the user data.

continued on next page

The Transport Layer *(continued)*

In the case of a send operation, the source address may be implicit, and in the case of a receive operation the destination address may be implicit. The QOS parameter is unspecified, as we discussed earlier. The user data parameter is just what it suggests: the data the transport user wishes to convey to its peer.

The OSI connectionless-mode service is similar to the service provided by the DARPA Internet protocol, User Datagram Protocol (UDP). The UDP service does not offer the quality of service parameter; otherwise the services are the same.

Connection-mode Service

The OSI connection-mode transport service offers many more guarantees than the connectionless-mode service. It guarantees that all data will arrive at their destination intact, uncorrupted, and in the same order in which they were sent; if this cannot be achieved the user of the transport service will be notified of the failure. In order to maintain the order of the data sent, the packets must be labelled in some way. In theory this can be done by the transport user or by the transport provider. The OSI Transport service does this labelling to preserve the order of the data, and in order to do so, it requires some context in which to label the data. This context is an association between the source and the destination transport users, called a transport connection. There is a phase of the connection-mode service in which the connection is established, a phase during which all data is transferred, and a phase during which the connection is dissolved. During the data transfer phase, the data is associated with a connection rather than with a pair of addresses, as in the connectionless-mode transport service. There may exist any number of connections with a given pair of transport addresses.

The service primitives for the connection-mode transport service are many and varied. Below they are described in the context of the phases to which they apply.

Connection establishment phase

It is easiest to describe the connection establishment phase with an example: a client process C, using the transport service provider TC on host HC, and a server process S, using the transport service provider TS on host HS. The primitives used for connection establishment are:

`T-CONNECT.request` (with parameters *called address, calling address, service options, quality of service*, and *user data*),

`T-CONNECT.indication` (*called address, calling address, service options, quality of service, user data*),

`T-CONNECT.response` (*quality of service, responding address, service options, user data*), and

`T-CONNECT.confirm` (*quality of service, responding address, service options, user data*). The client process, C, issues a `T-CONNECT.request` to TC. TC, the provider, sends a transport connection request TPDU to its peer, TS. TS issues a `T-CONNECT.indication` to S. The server process S gives permission to establish a connection by issuing a `T-CONNECT.response`. After a suitable protocol exchange between TS and TC, the client is informed of the successful connection establishment by means of a `T-CONNECT.confirm` primitive issued by TC.

Despite the different names, the *responding address* parameter on the `T-CONNECT.response` must match the called address on the `T-CONNECT.indication`. The *service option* parameter is used to request use of the expedited data service, which we describe below in the context of the data transfer phase.

The service modelled by these primitives stands in contrast to that provided by the TCP. Whereas the OSI Transport protocol uses a model that requires that, for each connection request, the provider obtain the user's explicit permission to establish a connection, the TCP provides a different model for connection establishment. The TCP model can be characterized as a passive-active model. If the Transport protocol were replaced by the TCP, the above scenario would change: S issues a primitive to give permission for TS to accept connection requests to a given destination address. The TCP service permits S to give "unspecified" permission, meaning any connection request to this destination address will be accepted, or to give "fully specified" permission, meaning that only connection requests from a given source address will be accepted. When C issues a `T-CONNECT.request`, a protocol exchange between TC and TS results in TS enqueueing a connection for S. S then issues a primitive to dequeue pending connections that have been enqueued in TS. Another protocol exchange occurs between TC and TS, and both TS and TC issue `T-CONNECT.confirm` primitives to S and C, respectively.

The TCP also permits the establishment of a single transport connection between two users that issue active open requests. The OSI Transport protocol does not. Simultaneous symmetric connection requests in the OSI scenario will result in two transport connections, unless, of course, one of the user processes recognizes the duplication and refuses the second connection.

Connection release phase

A (potential) transport connection is said to be released when a) it is destroyed by a user of the transport service, for example, when a user is finished with the transport connection; b) it is destroyed a provider of the transport service because of an inability to maintain the the connection, such as the failure of a network link; c) it fails to be established due to a provider's inability to establish the connection, for example, a provider's lack of memory buffers, or d) it cannot be established because a user of the the transport service rejects a connection request.

Two primitives apply to the connection release phase, namely: `T-DISCONNECT.request`, and `T-DISCONNECT.indication`.

Unlike the service provided by the TCP, the OSI Transport service does not provide a negotiated release of transport connections or a "graceful" close. The OSI Transport connection release service is similar to the TCP connection abort service.

The OSI Transport guarantee that data will be delivered means that data will be delivered to the transport *provider* at the destination. This does not necessarily mean that the transport service user will receive the data. The data may be buffered by the destination transport provider awaiting receipt by the destination user. If at this time the sending transport user dissolves the transport connection, the buffered data may be discarded.

continued on next page

The Transport Layer *(continued)*

This is in sharp contrast to the service provided by TCP, which offers the user the option of maintaining the connection until the buffered data have been received by the destination transport user. (TCP also provides an abort service primitive which, like the OSI Transport service, dissolves the connection immediately.) The TCP service is similar to the OSI Transport service in most other respects.

Data transfer phase

The OSI data transfer phase offers two services: the transfer of "normal data", and the transfer of "expedited data." In the following discussions, "data," when unspecified, means "normal data." First we will discuss normal data. The normal data transfer service uses two primitives: `T-DATA.request` (*user data*) and `T-DATA.indication` (*user data*). User data is transferred in packets or records, in OSI called TSDUs. There is no limit to the size of a TSDU. The TCP data transfer service does not provide record boundaries, but instead transfers a continuous stream of bytes.

Both the TCP service and the OSI Transport service provide, in addition to normal data transfer, features that allow the user to mark certain data as special, but there the similarity ends. The OSI *expedited data* service is an optional service, both in the sense that it is not provided by all classes of the Transport protocol, and also in the sense that the user may elect to prohibit the transfer of expedited data. The appellation "expedited data" is to some extent a misnomer. The service provides a separate data channel from the normal data channel. The "expedited" channel is not guaranteed to deliver data faster than the normal channel; it does not necessarily allow its contents to bypass data in the normal channel. It guarantees only that data placed in the normal data channel cannot bypass data in the expedited data channel. This has the result that the expedited channel can actually impede the delivery of normal data. The expedited data channel has very small bandwidth: an expedited TSDU (XSDU) may be at most 16 bytes, and at most one XSDU may be in the channel at any time. It is not intended for the transfer of bulk (or even moderate amounts of) data. Its most plausible use is to deliver information about exceptional conditions.

The analogue in the TCP service is called *urgent data*. Urgent data is embedded in the normal data stream; the TCP does not provide a second data channel. The TCP user is given an indication that it should enter "urgent mode," meaning that it should quickly process all the data up to a certain point in the data stream. When that point in the data stream is reached, the TCP user can be so informed, although the TCP standard does not specify how this is done.

Services not provided

TCP offers another feature for which there is no analogue in the OSI Transport service: *data stream push*. This feature permits the user to force the transport provider to move all data that has been buffered at the source to the destination provider and through the destination provider to the peer user. There is no way for the destination user to detect the receipt of "pushed" data, so this feature is not a substitute for a packet boundary marker (end-of-TSDU indicator) or for a user-level acknowledgement.

The pushed data is subject to the normal flow control restrictions of the protocol, so the push is no more than a hint to the provider to indicate to the provider not to wait for more data from the user before placing the data in TCP data packets and sending them to the destination provider.

Neither the TCP nor the OSI Transport service offers a user-level acknowledgement. The sending user can never tell when its peer has received any particular datum unless these peers implement a higher-layer protocol above the Transport Layer.

Neither the TCP nor the OSI Transport service offers multi-way transmission. If this service is required in either environment, it must be built in a higher layer protocol.

Neither the TCP nor the OSI Transport service provides a reliable connectionless service. Users of the connectionless transport service must build acknowledgments into a higher layer protocol if they need reliability without the overhead of a connection-mode protocol.

The protocols

In the rest of this article we describe the OSI Transport protocols and attempt to give a teleological treatment of their features. The OSI Transport Layer comprises six different protocols: the connectionless transport protocol and classes 0, 1, 2, 3, and 4 of the connection-mode transport protocol. Despite the nomenclature, classes 0 through 4 are five different protocols. They share a set of TPDU headers or at least, a common set of codes identifying the use of the TPDUs. In particular, they share the TPDUs used to initiate the connection establishment phase: the connection request (CR) TPDU and the connection confirm (CC) TPDU. Using these two TPDUs, the transport providers negotiate the protocol class, and hence, the format and set of TPDUs to be used for the rest of the connection.

The reason that there are five connection-mode protocols is that there is a variety of network services available. These network services provide varying degrees of reliability at varying costs, with a variety of charging algorithms. If use of the network incurs a charge for each bit or byte transferred, a transport protocol should minimize the amount of control information it uses, possibly at the expense of other considerations such as CPU processing overhead. On the other hand, if the cost of using a network is fixed, the transport protocol can maximize reliability or minimize CPU processing overhead at the (non-)expense of transferring extra bytes. The OSI Transport protocol classes are summarized as follows:

TP0

The *class 0* protocol is extremely simple. It is meant to be used with a network service that provides most or all of the functionality that the connection-mode transport service provides (expedited data transfer excepted). Class 0 incurs minimum additional expense and provides nothing more than transport level addressing. It is compatible with the CCITT Teletex protocol.

TP1

The *class 1* protocol is also very simple. It is meant to be used with a network service that provides most the functionality that the connection-mode transport service provides, but which may signal certain types of network errors and close the network connection. Class 1 recovers from these errors by opening a new network connection and resynchronizing the state of the two peer providers.

continued on next page

The Transport Layer *(continued)*

TP2 The *class 2* protocol does not provide the error recovery that class 1 provides. Class 2 requires a network service that is as reliable as that used by class 0. Class 2 provides multiplexing of transport connections over network connections to reduce the cost of using the network. It also provides, as optional capabilities, flow control at the Transport Layer, and a transport expedited data service.

TP3 The *class 3* protocol combines the error recovery of class 1 with the multiplexing, optional flow control, and expedited data service of class 2. All the classes up to class 3 require that the network service be a connection-mode service.

TP4 The *class 4* protocol makes no assumptions about the reliability of the network service. The class 4 protocol includes all the error detection and recovery and flow control functions required to offer the fullest connection-mode transport service defined by the service standard. It can do so using any type of network, reliable or unreliable, connection-mode or connectionless.

Several factors determine which transport class is to be used for a given instance of communication. These include: the choice of networks that can physically connect the source host with the destination host, what quality of service the user requires, the set of protocol classes implemented at the destination, and the accounting requirements of the system. For example, if the destination is reachable only over a connectionless network service, class 4 of the Transport protocol must be used because it is the only class that can run over a connectionless network service.

Class and option negotiation

The transport class is negotiated during connection establishment. The initiating provider proposes a preferred class to use and a list of acceptable alternatives. The responding provider chooses a class according to a complex set of negotiation rules. In essence the class is negotiated *down* from the preferred class. This is problematic because the initiator may propose a class that is too low for the responder. The responder may perceive its network service to be too error-prone for use with the proposed class, yet the responder cannot respond with a higher, more tolerant, class.

In addition to the negotiation of protocol class, other protocol options are negotiated during connection establishment. Negotiated options include such things as TPDU format (in classes 2, 3, and 4 there are two formats, one that accommodates a 7-bit sequence space and one that accommodates a 31-bit sequence space), the use of explicit flow control in class 2, the maximum size of TPDUs to be exchanged during the data transfer phase, the use of a network expedited data service, the use of a network receipt confirmation service (end-to-end acknowledgements), the use of checksums in class 4, the provision or denial of transport expedited data service for the connection, and the quality of transport service to be supported. This negotiation is further complicated by the fact that some options apply only in certain classes, and the classes that can be used are determined by, among other things, the network connectivity, hence by the network service to be used. This is not an issue for systems that have only one network service available, but for systems in an internetwork environment, particularly where the internetwork protocol (ISO 8473) is not ubiquitous, this is likely to be an issue.

Timers

An additional problem encountered during connection establishment is that of initial timer values. The connection establishment model suggests that the typical implementation of Transport will, after receiving a connection request, wait for approval from the application level before responding with a connection confirm. This may cause a significant delay in the connection confirmation. How does an initiating transport provider guess how long to wait before retransmitting its connection request, and how long to wait before giving up trying? If the timers are too short, a connection will never be established. If the timers are too long, a user may wait intolerably long before finding out that the destination is unreachable. There is some discussion in ISO to attempt to alleviate this problem by acknowledging receipt of a connection request without confirming it. Note that the passive-active model used by TCP does not have this problem—once a server initiates a passive open, TCP will automatically establish a connection on the server's behalf.

Data transfer

Data transfer in the simpler classes is trivial. In these classes, all essential service is provided by the network Layer. In the higher-numbered classes, that run over less reliable network layers, the protocols use mechanisms to provide reliable, flow controlled, sequenced data transfer service. First, TPDUs are numbered with a *sequence number*. Packets that arrive at the destination out of sequence are reordered so they are delivered to the user in the correct order. The two peer transport providers keep each other apprised of the data received by exchanging control messages (acknowledgments). The sending transport provider retains a copy of all data that was sent until it receives an acknowledgement that the data arrived at the destination. This acknowledgement mechanism is also used to effect *flow control*, which prevents the sender from sending more data than the receiver can store in its memory buffers. This is accomplished by exchanging *credit*. Credit is an indication of the size of the *window*, which is the set of TPDUs a receiver can still receive without running out of memory.

Since the Transport Layer manages many transport connections simultaneously, it is unwise for the transport provider to permit one transport connection to consume all the available memory. The *buffering strategy* used by a transport protocol implementation is a critical factor in the performance of the protocol. It must permit *fairness* among transport connections and *high throughput* for connections with a variety of usage patterns. Even though the buffer strategy is not a part of the protocol specification, designing a good memory management (buffer) service for the protocol is perhaps the central problem in protocol implementation.

Inactivity control

Class 4 contains a mechanism for closing a connection when it appears that the Network Layer connectivity between the peers has been compromised. This is called *inactivity control*. It is needed because the class 4 provider cannot use the closing of a network connection to indicate that it should consider the connection closed. Remember that class 4 makes no association between network and transport connections, and in fact, may run over a connectionless network service. Nor can the class 4 provider hold a transport connection open until it receives a disconnect request TPDU, since the network service over which class 4 runs may be unreliable, and may lose disconnect requests. Inactivity control requires that the peer providers exchange acknowledgments regularly.

continued on next page

The Transport Layer *(continued)*

If a period of time, called the *local inactivity time*, passes without receipt of anything from its peer, a transport provider considers the transport connection to be released. The length of the inactivity time is a local matter, and so is the amount of time a provider chooses to wait between sending acknowledgments on an otherwise quiescent transport connection (which we will call the *keepalive time*). Obviously, if one provider has an inactivity time that is shorter than its peer's keepalive time, the connection may be closed prematurely. This problem is alleviated by the *flow control confirmation* mechanism, described below.

The OSI Transport protocol contains some complex mechanisms for maintaining a consistent view of the data transfer state between peer providers when they are using a network service that loses or reorders packets. These mechanisms are *acknowledgement subsequencing* and *flow control confirmation*. The former consists of numbering acknowledgement TPDUs so that a receiving transport provider can tell when acknowledgments arrive out of order. This is meant to prevent inadvertent retransmission and possibly closing of the connection that can occur if acknowledgements arrive in an order different from the order in which they were sent.

Flow control

Flow control confirmation accomplishes three things: it verifies that credit that was once reneged has been reissued, it verifies that a closed window has been opened, and it synchronizes inactivity control. This mechanism calls for adding a parameter to an acknowledgement. The parameter contains information from the most recently received acknowledgement. The protocol standard does not dictate the actions of a provider that receives flow control confirmation information that is inconsistent with its own flow control information, but the logical thing to do is to send another acknowledgment. The flow control mechanism is defined such that when a transport connection is quiescent (no data is being transferred), acknowledgements that are sent for the purpose of inactivity control will cause the peer to send an acknowledgement back. This synchronizes inactivity control in both directions, and prevents the premature release of a connection *provided that* each provider has an inactivity time that is sufficiently longer than its keepalive time. "Sufficiently longer" means longer by at least the round trip time.

Error detection and correction

The last protocol mechanism that we would like to discuss in this article is the *error detection and recovery* mechanism. Both the TCP and the OSI Transport protocols use checksumming to detect end-to-end errors. The Fletcher checksum algorithm is used for the OSI Transport protocol. While it is slightly stronger than the checksum algorithm used for the TCP, it takes up many more CPU cycles. Whereas the TCP checksum algorithm requires one addition for each two bytes of data, the Fletcher algorithm requires two additions for each byte of data. Because of the high cost of using checksumming in the OSI Transport protocol, its use is subject to negotiation during connection establishment.

Summary

The OSI Transport Layer provides a rich set of features and two models of data transfer service: reliable, sequenced connection-mode, and datagram mode. These two models are roughly equivalent to the services provided by the DARPA Internet protocols TCP and UDP, respectively.

Each of these protocols offers one or more service features that is not provided by the other. The OSI Transport protocol contains some fairly complex mechanisms intended to permit the Transport Layer to adapt to the type and quality of network services on which it runs and to maintain a high-quality transport service regardless of the characteristics of the underlying network. This can cause tremendous complexity in implementations owing to the large number of options and exceptions possible. In contract, by assuming a worst-case network service, and using a common network protocol (IP), TCP avoids these problems while achieving equivalent functionality.

Bibliography

ISO 8072, Transport Service Definition.

ISO 8072/AD1, Addendum 1: Connectionless-mode Transmission.

ISO 8073, Transport Protocol Specification.

ISO 8073/AD1, Addendum 1: Network Connection Management Subprotocol (NCMS).

ISO 8073/AD2, Addendum 2: Class 4 Operation over Connectionless Network Service.

ISO 8602, Protocol for Providing the Connectionless-mode Transport Service.

Board on Telecommunications and Computer Applications, Commission on Engineering and Technical Systems, National Research Council, "Transport Protocols for Department of Defense Data Networks," 1985.

Bricker, A., Landweber, L., Lebeck, T., Vernon, M., "ISO Transport Protocol Experiments," Mitre Technical Report MTR86W00002, 1986.

Fletcher, J., "An Arithmetic Checksum Algorithm for Serial Transmission," *IEEE Transactions on Communications*, COM-30, January, 1982.

Nakassis, A., "Fletcher's Error Detection Algorithm: How to implement it efficiently and how to avoid the most common pitfalls," *Computer Communication Review*, October 1988.

Cockburn, A., "Efficient Implementation of the OSI Transport Protocol Checksum Algorithm Using 8/16-Bit Arithmetic," *Computer Communication Review*, July/August 1987.

NANCY E. HALL received an MS in Computer Sciences from the University of Wisconsin-Madison in 1977. She is an Associate Researcher in Network Communications at the University of Wisconsin-Madison. Her current work involves implementation of the ISO Network and Transport protocols and UNIX kernel systems support for the Wisconsin ARGO project (ISO protocol implementation for the IBM PC RT workstation in a Berkeley UNIX environment). She is a member of the ANSI X3S3.3 standards development group. From 1982 to 1985 she worked on the Crystal multicomputer project including implementation of a Modula compiler, maintenance for network software and design and development of user utilities. From 1977 to 1982 Ms. Hall worked as a systems programmer for the University of Wisconsin Academic Computer Center. She also taught courses on data structures, languages, and utilities.

Interoperability Testing: The Final Test of Open Systems

by Marshall T. Rose, NYSERNet, Inc.

The goal of Open Systems is to have true interoperability between the products of computer communications vendors. When interoperability is achieved, then the user is able to leverage the potential of Open Systems: the best tool can be selected for the job.

Standards as the vehicle towards Open Systems

OSI is produced through a series of successive refinements: *Standards* are produced by a standards body, such as the International Organization for Standardization (ISO). These standards are taken by profiling organizations, such as the OSI Implementor's Workshop sponsored by the National Institute of Standards and Technology (NIST), and *functional profiles*—implementor's agreements of the standards—are defined. Finally, an organization takes one of these functional profiles and declares it to be a *standard profile*, as a part of their procurement policy. For example, FIPS 146, more commonly known as the U.S. Government OSI Profile (US GOSIP), mandates the future use of GOSIP-compliant OSI products by federal users.

An open question (no pun intended) is how one ensures interoperability among OSI products, especially when separately developed by different vendors. The Standards define the base technology and the Profiles define subsets of the Standards and provide clarifying information. But, how can the user be sure that the products actually interoperate?

Although there are several possibilities, many feel that pair-wise interoperability testing is the litmus test of OSI. Many believe that interoperability testing, and nothing else, provides the final proof.

Testing as the Proof of Open Systems

In order to appreciate the value of interoperability testing, it is helpful to consider how it has proved successful for OSI's predecessor, the Internet suite of Protocols, commonly referred to as TCP/IP.

Early in the development of the Internet suite of protocols, an interoperability "bake-off" was hosted at the USC/Information Sciences Institute. Ten different implementations were represented. The test began by having all implementations connected to the same network with the *checksumming* algorithm disabled. This algorithm ensures data reliability by computing an arithmetic sum, the checksum, over the data sent. The receiver looks at the data and the checksum and determines if either has been corrupted during the transmission.

With the checksumming algorithm disabled, all the bake-off implementations were able to interoperate. Another round of tests were run, this time with the checksumming algorithm enabled. Only two implementations could interoperate! The two implementors were asked to explain their interpretation of the checksum algorithm. The document describing the Internet checksumming algorithm was then modified to contain these explanations, and the other implementors changed their code. The results became the standard way for calculating Internet checksums.

This amusing anecdote illustrates several interesting things about both interoperability testing and the Internet suite. In particular, even though the original Internet checksum algorithm was fairly well specified (ten different groups were able to implement it), it was sufficiently ambiguous to result in many implementations which did not interoperate, yet had reason to believe that they were conformant.

Interoperability testing has served the Internet suite well: today there are over a hundred vendors offering thousands of TCP/IP products which interoperate. Little formal testing was done, the majority of testing occurred as new products were introduced to an existing environment: the true test of Open Systems.

OSI needs Interoperability Testing

In comparison, the Internet protocols are trivial with respect to OSI. As such, the possibility of subtle interactions between implementations or of misinterpretation of the Standards or Profiles is much greater.

At present, the majority of all OSI interoperability testing occurs when an OSI trade show is announced. For example, in March, 1989, the EurOSInet organization sponsored a multi-vendor OSI booth at the CeBIT '89 show in Hannover, FRG. A few months before the show, the technical representatives began coordination for interoperability testing.

The interactions can be broken down into two categories: testing between systems which had already been tested at previous shows, and testing between systems that had never been tested with each other. In the first case, testing was largely a formality. In the latter case, testing was an absolute disaster for the technical personnel involved!

It is important to appreciate that while hundreds of vendors have announced OSI capabilities of one form or another, there are probably less than ten completely independent implementations. When an independent implementation begins testing for the first time, it is almost certain to unintentionally crash each and every one of the different implementations it tests against!

Here's why: the most complicated aspect of OSI is the negotiation of *options* at the beginning of a communication. There are always many options to be negotiated (regardless of functional profiles), and many facilities to be proposed, and then either enabled or disabled. The new implementation will likely have some bugs in the options it proposes. However, in the majority of cases it will probably be the case that the earlier implementations simply don't handle option negotiation gracefully. Interestingly enough, the other implementations probably crash on different options and at different points in execution. After all, since they are different implementations, they handle options differently and with different levels of resilience.

The following scenario is entirely true. Consider one of many events that occurring during testing for the EurOSInet demonstration: in testing a new implementation of FTAM (the OSI File Transfer, Access and Management Standard) against an "established" implementation (one which had successfully tested against other implementations), the older implementation would mysteriously crash in the middle of a transaction.

continued on next page

Interoperability Testing *(continued)*

After more than a man-month of investigation, a bug in the newer implementation was found: it did not propose a mandatory parameter. Once this was fixed, communications occurred without any further problems. Although the bug was in the newer implementation, this example points to a *larger* problem in the older implementation: it simply isn't robust. The lack of a mandatory parameter should have immediately set off alarm bells in the older implementation. It did not; instead, at some indeterminate time later, the older implementation would crash, for no apparent reason! Interestingly enough, none of the other implementations detected the missing parameter, or crashed; they were robust enough to cope with the situation that arose after negotiations were completed.

Of course, as more implementations of OSI are developed, the problem of interoperability testing will grow larger. It is ironic that the new implementations, while helping to fulfill the promise of OSI, also contribute to the amount of effort required to achieve Open Systems.

The role of Conformance Testing

Given this perspective, what role does conformance testing play? Many view a conformance test as a kind of interoperability test: the *implementation under test* is asked to interoperate with the *conformance tester implementation*. The difference is that the conformance tester may purposely "act strangely," perhaps violating the protocol specifications, so as to provoke a response from the implementation under test. This serves to test for robustness.

Obviously, conformance testing is useful when an implementation is being developed—it provides a sanity checking tool for the implementation under development. But, can it substitute for pairwise interoperability testing? The problem is that it is difficult to anticipate the problems encountered when independent implementations interact. Unless the conformance tester can emulate the behavior of all other implementations, then it can not replace pairwise interoperability testing. As an example, all aspects of timing and error conditions can not be anticipated or enumerated—the domain is simply too large.

Thus, while conformance testing can give some additional confidence to developers, it is paradoxical that it offers little to users. Users run *products*, not conformance testers, in their networks. They need interoperability testing between vendor products, not interoperability testing against a conformance tester implementation.

On the Horizon

An excellent example of the state of the art in interoperability testing will be the INTEROP™ 89 show in San Jose, California, October 2–6. (See page 15) This is a network exhibition and technical conference sponsored by Advanced Computing Environments. This year's event will be the 4th TCP/IP interoperability event. Each year, participation grows: last year's event, the first having a network exhibition, demonstrated interoperability between 50 vendors to over 5400 users. This year even larger participation and attendance is expected.

With OSI following in TCP/IP's footsteps as the standard of choice for open networking, we should look forward to OSI-based INTEROP events in the years to come!

Special Demonstrations for INTEROP™ 89

As was the case with last year's INTEROP Exhibition, we are planning a number of cooperative demonstrations between different manufacturers who have implemented common products or protocols. These demonstrations are intended to show attendees that the standards they read about in technical journals are a reality.

If your company is exhibiting and has an interest in participating in these activities, please contact Peter de Vries of Advanced Computing Environments at 415-941-3399. He will be acting as liaison between vendors who are preparing the demonstrations. For your reference, here are some of the current plans for cooperative demos at INTEROP 89:

CMOT

Since the original NetMan demonstration last year, RFC 1095 has been issued as a recommended standard for CMIP over TCP/IP (CMOT). This year, the vendors who have developed CMOT-compliant software for their systems will be showing products which implement the CMOT network management architecture. Attendees will be able to see current packages which conform to this specification.

SNMP

An SNMP Demonstration is planned which will display the current solutions offered by the SNMP network management utility suite as it has matured over the course of the first year of its RFC's life span. This protocol has gained importance as increasing numbers of vendors have implemented it in the marketplace and are now refining its usefulness to network customers.

TCP/IP over FDDI

Another group of vendors is working on a demonstration of TCP/IP over FDDI (Fiber Distributed Data Interface). This will show attendees another page from the future of high-speed local area networking—a token-passing glass fiber ring that will connect systems (and LANs) together in the years to come. Applications based on TCP/IP will be exhibited on a network of FDDI nodes. This will be the first public demonstration of a multi-vendor FDDI network using TCP/IP as its protocol standard.

OSI Networking

As the OSI protocol standards are getting closer to being finalized, more vendors are completing their ports of the lower levels of the protocol. In point of fact, some companies' product lines are focused *entirely* upon the ISO/OSI architecture. These vendors will have their software on display, while several of the other booths will be communicating via both OSI- and TCP/IP stacks.

NetBIOS over OSI

The NetBIOS Special Interest Group will present a unique demonstration of OSI-based multi-vendor NetBIOS interoperability. The demonstration will consist of a fully-functioning OSI network complete with some of today's most popular software operating via a standard NetBIOS interface over the lower layers of the OSI protocol stack. This will demonstrate how it will be possible for a site to retain and use transparently most existing applications while starting the migration to OSI from either TCP/IP or XNS.

X Window System

Using a foundation of multiple servers and clients residing on a heterogeneous group of machines, vendors will be displaying the latest in distributed applications for the X Window System. In addition, real-time animation will be possible, for the first time, using INTEROP's high-speed, fiber-based backbone. —*Peter de Vries*

Recent Publications

NMSL

Specification and Verification of Network Managers for Large Internets, TR 832, by A David L. Cohrs, A Barton P. Miller, University of Wisconsin. Abstract: Large internet environments are increasing the difficulty of network management. Integrating increasing numbers of autonomous subnetworks (each with an increasing number of hosts) makes it more difficult to determine if the network managers of the subnetworks will interoperate correctly. We propose a high level, formal specification language, NMSL, as an aid in solving this problem. NMSL has two modes of operation, a descriptive mode and a prescriptive mode. In its descriptive mode, NMSL specifies abstractions for the network components and their instantiations, and verifies the consistency of such a specification. The abstractions include the data objects and processes in a network management system. These abstractions are instantiated on network elements and are grouped together in the specification of domains of administration. An extension mechanism is provided to allow for the specification of new management characteristics that the basic language cannot express. In its prescriptive mode, NMSL generates configuration information directly from a consistent specification. This information is used to configure network management processes to make their operation consistent with their specifications. Standard management protocols (such as the emerging ISO or IETF standards) can be used to incorporate the configuration information into running management processes. The report may be ordered from:

Technical Report Librarian
Computer Sciences Department
University of Wisconsin
1210 W. Dayton Street
Madison, WI 53706

Security Problems in TCP/IP

Security Problems in the TCP/IP Protocol Suite, by Steven Bellovin, AT&T Bell Laboratories. Published in ACM SIGCOMM *Computer Communications Review*, (CCR) Volume 19, Number 2, April 1989. Abstract: The TCP/IP protocol suite, which is widely used today, was developed under the sponsorship of the Department of Defense. Despite that, there are a number of serious security flaws inherent in the protocols, regardless of the correctness of any implementations. We describe a variety of attacks, source address spoofing, and authentication attacks. We also present defenses against these attacks, and conclude with a discussion of broad-spectrum defenses such as encryption. [Ed.: Also in the same issue of CCR is a table entitled "Status of OSI (and related) Standards."]

Congestion Avoidance

DEC-TR-566: A Delay-Based Approach for Congestion Avoidance in Interconnected Heterogeneous Computer Networks by Raj Jain, DEC. Abstract: In heterogeneous networks, achieving congestion avoidance is difficult because the congestion feedback from one subnetwork may have no meaning to sources on other subnetworks. We propose using changes in round-trip delay as an implicit feedback. Using a black-box model of the network, we derive an expression for the optimal window as a function of the gradient of the delay-window curve. The problems of selfish optimum and social optimum are also addressed. It is shown that without a careful design, it is possible to get into a race condition during heavy congestion, where each user wants more resources than others, thereby leading to a diverging condition.

It is shown that congestion avoidance using round-trip delay is a promising approach. The approach has the advantage that there is absolutely no overhead for the network itself. It is exemplified by a simple scheme. The performance of the scheme is analyzed using a simulation model. The scheme is shown to be efficient, fair, convergent, and adaptive to changes in network configuration. The scheme as described works only for networks which can be modeled with queuing servers with constant service times. Further work is required to extend it for implementation in practical networks. Several directions for future research have been suggested.

Caching Schemes

DEC-TR-592 Characteristics of Destination Address Locality in Computer Networks: A Comparison of Caching Schemes by Raj Jain, DEC. Abstract: The size of computer networks, along with their bandwidths, is growing exponentially. To support these large, high-speed networks, it is necessary to be able to forward packets in a few microseconds. One part of the forwarding operation consists of searching through a large address database. This problem is encountered in the design of adapters, bridges, routers, gateways, and name servers. Caching can reduce the lookup time if there is a locality in the address reference pattern. Using a destination reference trace measured on an extended local area network, we attempt to see if the destination references do have a significant locality. We compared the performance of MIN, LRU, FIFO, and random cache replacement algorithms and found that the interactive (terminal) traffic in our sample had quite different locality behavior than that of the noninteractive traffic. The interactive traffic did not follow the LRU stack model while the noninteractive traffic did. Examples are shown of the environments in which caching can help as well as those in which caching can hurt, unless the cache size is large.

Hashing Schemes

DEC-TR-593: A Comparison of Hashing Schemes for Address Lookup in Computer Networks by Raj Jain, DEC. Abstract: The trend toward networks becoming larger and faster, and addresses increasing in size, has impelled a need to explore alternatives for fast address recognition. Hashing is one such alternative which can help minimize the address search time in adapters, bridges, routers, gateways, and name servers. Using a trace of address references, we compared the efficiency of several different hashing functions and found that the cyclic redundancy checking (CRC) polynomials provide excellent hashing functions. For software implementation, Fletcher checksum provides a good hashing function. Straightforward folding of address octets using the exclusive-or operation is also a good hashing function. For some applications, bit extraction from the address can be used.

Getting DEC Technical Reports

If you would like to receive a copy of these reports, please send a message to jain%erlang.DEC@decwrl.dec.com with the subject field containing the word DEC-TR along with the report numbers you want. The message should be simply your US (or foreign) mail address such that it can be used directly on a mailing label. Any other words or sentences in the message will simply delay the delivery. You may also submit your request in writing to:

Raj Jain
Digital Equipment Corporation
550 King Street
Littleton, MA 01460

Book Review

Keeping The Link, Ethernet Installation And Management, by Martin Nemzow, 1988, McGraw Hill, Inc. 366 pages with index. Keeping the link could be quite difficult if you follow the advice given in this book. The book contains serious errors of fact with respect to Ethernet/802.3 standards. For instance, almost all of the thin Ethernet specifications given in the book are wrong. The book contains errors about higher level protocol usage as well, while other advice given in the book is at variance with standard industry practices. A few examples will suffice:

Thin Ethernet confusion

"Cheapernet and Thinnet, which run on the thinner 75 ohm cabling, support a maximum 200 meter segment..." (p 39)

The author apparently believes that there are two variants of the thin Ethernet technology, both of which run on 75 ohm cables. This belief is stated repeatedly throughout the book. While there are a number of vendor names for thin Ethernet, none of those products run on 75 ohm cable. There has never been a 75 ohm 10 megabit baseband Ethernet specification and the maximum segment length for 10BASE2 thin Ethernet is 185 meters.

"Baseband Ethernet coax supports a bus topology with a maximum circumference of a kilometer and a half, provides network access to a maximum of 1024 nodes (address space is limited to 10 bits)..."(p 41)

A given Ethernet can be 2800 meters long. It's hard to imagine how the author came up with a 10 bit field, unless he was inventing something in an attempt to explain the 1024 host limitation of Ethernet. There is *no* address field of 10 bits anywhere in the Ethernet specifications, and the 1024 host limitation has nothing to do with addressing fields.

Heartbeat

"When the network is not busy, an idle signal is transmitted by all source/destination nodes, which is the 0.7-volt carrier sense. This voltage is sometimes called the heartbeat." (p 55)

The drawings make clear that the author believes that the coax signals go from positive .7 volts to negative .7 volts. This is completely incorrect, as is the notion that the coax idle voltage is called heartbeat. The author appears to have confused the differential transceiver cable voltages with the coaxial cable voltages. Heartbeat is one name for a signal seen on the transceiver cable as well, although it has nothing to do with idle signals.

"DECnet is a TCP/IP implementation from Digital Equipment Corporation." (p 33)

This will come as a major surprise to the entire networking world!

One could go on, but the point is made. A list of errata for this book would be a large task. The author has committed to print a number of strange ideas that are completely at variance with the Ethernet specifications, not to mention the wider world of network protocols.

Pity the unwary reader of this book, since it is the case that using cables of the incorrect impedance for thin Ethernet may indeed work marginally for networks that are sufficiently short and sparsely populated.

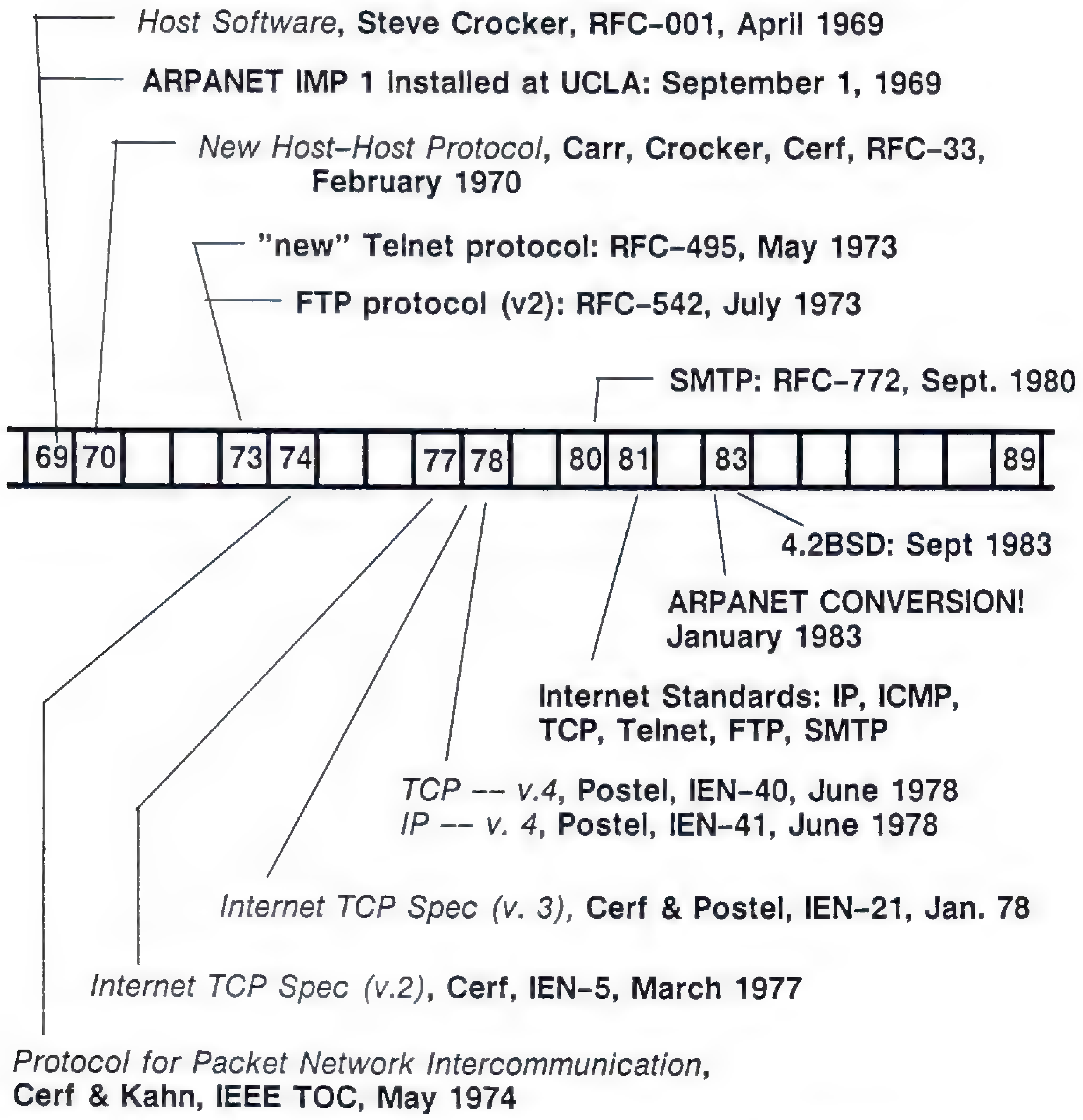
In actual practice any Ethernets built with incorrect coaxial cables will be highly error prone and can fail completely once they reach a certain traffic level and/or host population.

Not recommended

There are some sections of this book that verge on being useful, especially the later chapters on network construction and trouble-shooting. With some editing of redundant or useless tables and lists, they could be of assistance to a network manager. But there are so many incorrect statements made throughout the text that it is impossible to recommend.

It's sad to see this much effort go to waste. It's possible that if the book had received rigorous editing and if the material had been completely reviewed for technical accuracy it could have been salvaged. But as it is, this book is capable of leading people seriously astray. —*Charles Spurgeon*

SOME MILESTONES OF INTERNET HISTORY



Host Software, Steve Crocker, RFC-001, April 1969
ARPANET IMP 1 installed at UCLA: September 1, 1969
New Host-Host Protocol, Carr, Crocker, Cerf, RFC-33, February 1970
"new" Telnet protocol: RFC-495, May 1973
FTP protocol (v2): RFC-542, July 1973
SMTP: RFC-772, Sept. 1980
69
70
73
74
77
78
80
81
83
89
4.2BSD: Sept 1983
ARPANET CONVERSION! January 1983
Internet Standards: IP, ICMP, TCP, Telnet, FTP, SMTP
TCP -- v.4, Postel, IEN-40, June 1978
IP -- v. 4, Postel, IEN-41, June 1978
Internet TCP Spec (v. 3), Cerf & Postel, IEN-21, Jan. 78
Internet TCP Spec (v.2), Cerf, IEN-5, March 1977
Protocol for Packet Network Intercommunication, Cerf & Kahn, IEEE TOC, May 1974

As we approach the 20th anniversary of the ARPANET, it is interesting to look at the 20-year timeline and plot some major networking events. Bob Braden of USC-ISI did this recently for our Host Requirements Seminar and the result is shown above.

Workstation-Oriented Enhancements to Telnet

by James VanBokkelen, FTP Software, Inc.

Stuck in the middle

While many modern personal computers and workstations have bit-mapped displays, most applications software still expects to communicate with an asynchronous terminal. Mechanisms like BSD "termcap" and VMS Screen Management allow applications to support many kinds of asynch terminal, and workstation windowing software typically includes one or more asynch terminal emulations. The transition from simple terminal-style user interfaces to more sophisticated schemes is evident in many places, but it will take a long time to complete. Meanwhile, three recent RFCs describe additions to the Telnet protocol which make things easier on the users stuck in the middle.

Negotiate About Window Size option defined

RFC 1075, defines a mechanism to pass information about display window sizes and size changes from client to server (and to co-operating full-screen applications). A new Telnet option, Negotiate About Window Size (NAWS), is specified, and client and server are expected to negotiate it with `WILL/WONT/DO/DONT` like other options. If NAWS is agreed on, then the client can send Telnet Sub-Negotiation sequences to the server to indicate the initial display aperture dimensions (in characters), and further Sub-Negotiation sequences later if the dimensions change.

Two previously defined Telnet options, Negotiate Output Line Width and Negotiate Output Page Size, had never entered common use, and RFC 1075 comments that they were not adequate for this use, in any case. NAWS allows window dimensions up to 65536 x 65536, and establishes both dimensions in one operation. The older options were limited to 253 x 253, and required one operation per dimension changed. Since most window changes affect both dimensions, this might result in two successive screen updates being sent to the client.

Terminal Type option enhanced

RFC 1091 extends the existing Telnet Terminal Type (TTTYPE) option, to allow the Telnet server to review the client's list of supported terminal emulations, and select the one it likes best. Later, possibly at an application's request, another terminal type from the suported list can be selected.

As the option had previously been defined, in RFC 930, the client indicated end-of-list by sending the last supported type a second time. Thus, a server could only see the list of types once, and because the most desirable type was listed first, most existing implementations simply settled for it. Also, multiple emulation modes had not really been dealt with, and the exact point at which the mode changed was not specified.

The new specification builds on the old, while maintaining backwards compatibility with RFC 930. The option value remains the same, and old servers will receive the same end-of-list indication as before. However, if a new server sends one more request after receiving the end-of-list marker, a new client is expected to return to the top of its list. The mode change is defined as taking place when each new type is sent.

X Display Location Option defined

RFC 1096 defines a mechanism to pass the X-Window "display location" string across a Telnet connection, without explicit user intervention. A new Telnet option, X-Display-Location (XDISPLOC) is specified, and client and server are expected to negotiate its use as usual. If the client agrees to it, the server can send Telnet Sub-Negotiation sequences to query for an ASCII string of the format "host:display_num[.screen_num]." This is patterned on the format used for the UNIX DISPLAY environment variable, with an Internet host name pre-pended to it.

Prior to XDISPLOC, if a user at an X *server* (remember, the X protocol inverts the normal Internet sense of client and server: you type at the server end) wished to Telnet to a remote host and start an X client application, he would have to manually indicate which X display to use. This usually meant displaying the environment on his workstation, and typing in a fairly elaborate command line on the remote machine. With XDISPLOC, the Telnet server can query for it, and make it available to applications on the remote host, possibly as part of the environment. Starting an X client application becomes much simpler.

What the Network Maintainer can expect

When will this functionality become available in your own environment? That depends on several factors: The most important is the energy and level of interest of your vendors (or the authors and maintainers of any public-domain software in use at your site). A second factor is how useful each option might be in your environment. XDISPLOC is highly specific to X Windows; it won't serve any purpose unless a host can act as either an X client or server. NAWS is most important where windowing software is widely used, but it may see wider use to simply establish display dimensions at session startup time (e.g. when your display can be run as 24 x 80, 43 x 80 or 24 x 132). The extension to TTTYPE is most important in business and non-UNIX environments, where applications are more likely to be coded for use on only a few kinds of terminals.

So far, the first implementations of XDISPLOC and NAWS have been in public-domain software packages maintained at CMU. As of this writing, I know of two TCP/IP vendors (one primarily oriented towards VAX/VMS, the other towards PC/DOS) who are at work on support for the extended Terminal Type option. It is hard to predict how broadly accepted any of these options may become. Much depends on whether they are incorporated into future releases of widely used non-commercial TCP/IP implementations like the one distributed in 4BSD UNIX, or the *PC-IP* package developed by MIT, CMU, and Harvard, both of which have been extensively borrowed from by commercial vendors.

JAMES VanBOKKELEN's undergraduate relationship with MIT ended (scoreless tie) in 1980. From 1980 to 1985, he was Manager, Software Development for Perception Technology Corp., working on touch-tone data entry/voice response devices (does the IRS "Teletax" system ring a bell? No? Oh, well.) In 1986, he helped found FTP Software Inc., becoming the company's first VP of Marketing. Circumstances being what they are, he found himself getting sucked back into Software, replacing John Romkey as VP in 1987, and on the principle that the biggest pieces rise to the top, replaced Roxanne VanBokkelen as President in 1988. Most of his time is spent working on projects no one else here will touch, like the IETF Host Requirements Working Group, RFC 1001/1002 NetBIOS, TCP subtleties, and answering lots of e-mail.

New Telnet Working Group formed

by Dave Borman, Cray Research

The Internet Engineering Task Force (IETF) has formed a new working group to address the Telnet protocol.

In the last six months to a year, there has been a lot of interest in the Telnet protocol, a lot of new options have been proposed [Ed.: see preceding article], and a lot of discussion has gone into how to make Telnet better.

Purpose

The purpose of this working group will be to collect the current uses and wishes for Telnet, and produce one or more RFCs to address these issues.

One possible outcome from this working group would be to issue a new RFC on Telnet that would obsolete RFC 854. Also, some new Telnet options may be added to fix problems/limitations that people are currently having with Telnet (an option to allow automatic user authentication is needed).

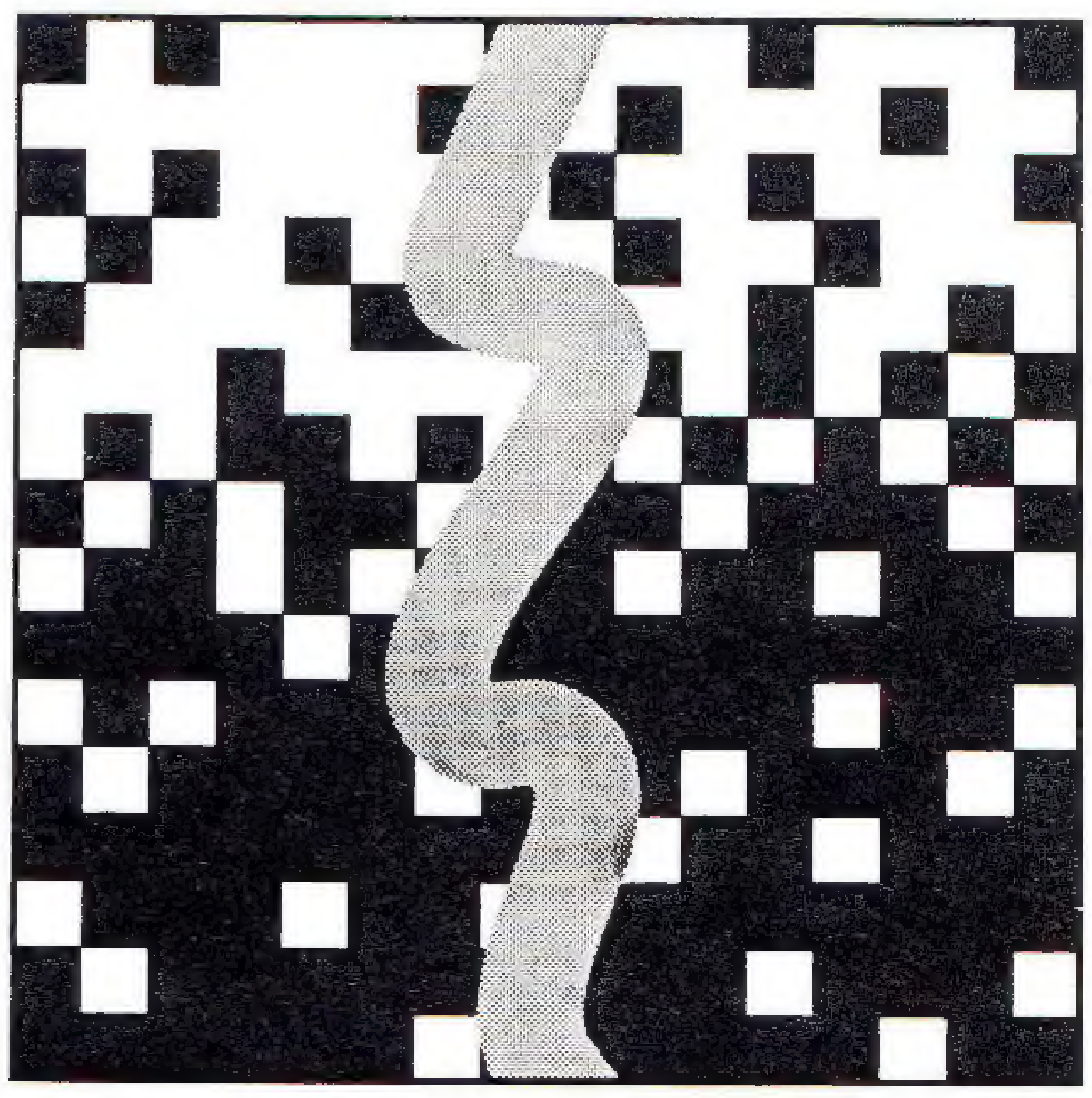
Meetings and mailing list

This group will have its first meeting at the next IETF meeting (at the end of July at Stanford). There will be a mailing list for this working group, and I hope that a lot of the discussion will happen over the mailing list, even before the first meeting of the group. Send me your name if you want to be on this mailing list. When the mailing list is formed, you will receive mail about where it is located. Also, please indicate whether or not you are involved in the IETF and would be at the next meeting.

Send your requests to dab@cray.com, or, if you don't have name-server and MX record support, dab%cray.com@uc.msc.umn.edu.

First International Symposium on Integrated Network Management held

The First International Symposium on Integrated Network Management, sponsored by IFIP and hosted by the MITRE Corporation and the National Institute of Standards and Technology, was held in Boston, MA, May 14–17, 1989. The audience of over 600 consisted of international industry and academic users, members of standards organizations, and hardware and software vendors.



An impressive array of technical and panel sessions, tutorials, guest speakers and vendor forums provided the attendees with a first hand view of what technology is currently available to manage diverse networks, and what is being developed by academic institutions and standardization bodies. The Symposium Proceeding are available from the North-Holland Publishing Company. The ISBN number is 0 444 87398 8.

Getting Back Issues

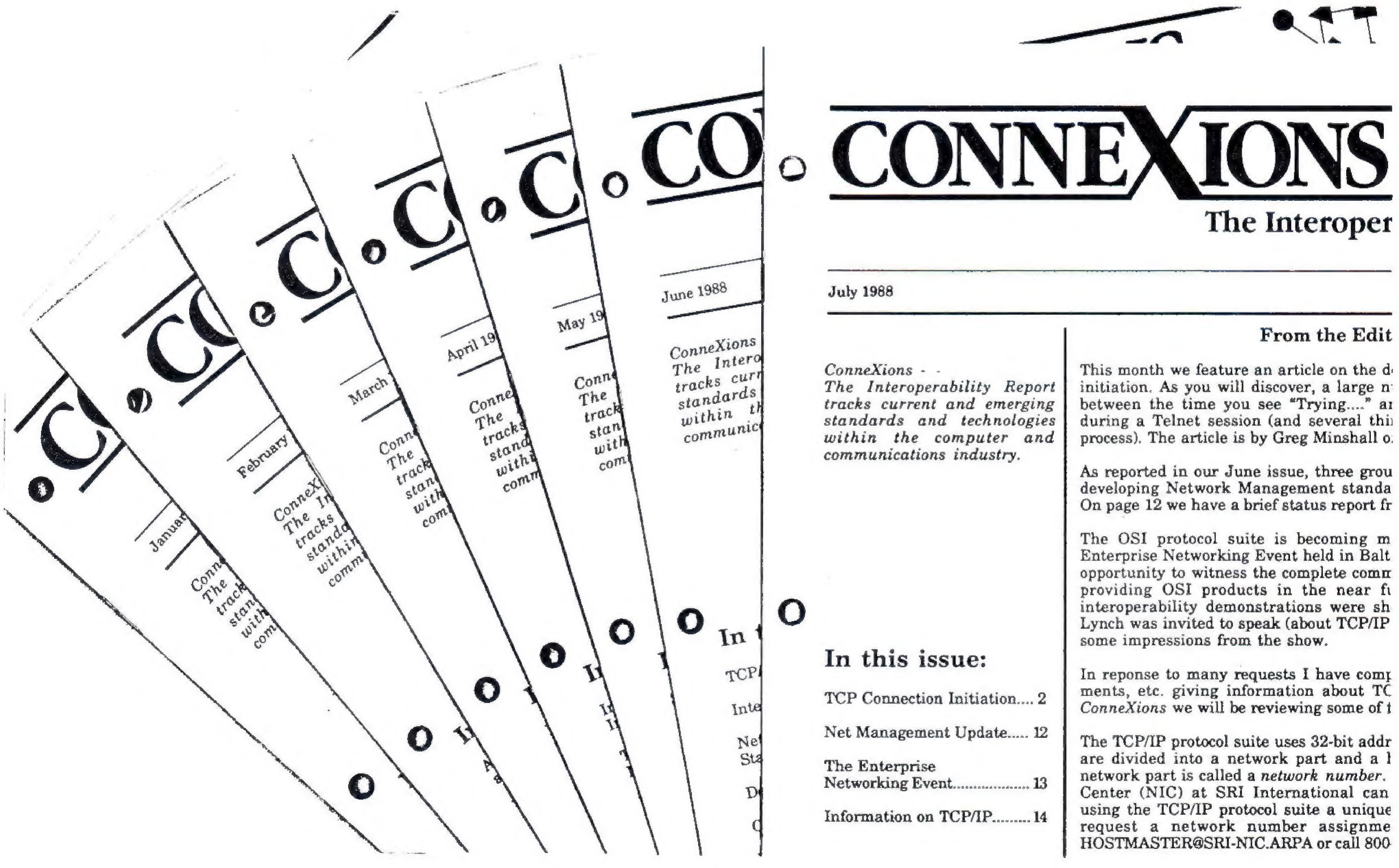
Every issue of *ConneXions* is produced with 3 holes punched in it so that you can put it in a standard binder.* We are also planning to produce a special *ConneXions* binder later this year. A complete set of back issues provides a great reference binder on TCP/IP and OSI interoperability topics.

Included with this issue is a Table of Contents for January–July 1989. Tables for Volume 1 (1987) and Volume 2 (1988) are also available free of charge.

You can order any back issue for $15, including shipping and handling. A special offer of $100 for 21 back issues is also available. To obtain back issues, simply specify which issues you require and send your order to Advanced Computing Environments, 480 San Antonio Road, Suite 100, Mountain View, CA 94040. Call us at 415-941-3399 for further information.

To date, there have been 3 special issues: *Protocol Testing* (Volume 2, No. 8, August 1988), *Subnets* (Volume 3, No. 1, January 1989) and *Network Management* (Volume 3, No. 3, March 1989).

For general subscription information, see the back page.



CONNEXIONS

The Interoper

July 1988

ConneXions - - The Interoperability Report tracks current and emerging standards and technologies within the computer and communications industry.

In this issue:

TCP Connection Initiation.... 2

Net Management Update..... 12

The Enterprise Networking Event.................. 13

Information on TCP/IP......... 14

From the Edit

This month we feature an article on the d initiation. As you will discover, a large n between the time you see "Trying...." a during a Telnet session (and several thi process). The article is by Greg Minshall o

As reported in our June issue, three grou developing Network Management standa On page 12 we have a brief status report fr

The OSI protocol suite is becoming m Enterprise Networking Event held in Balt opportunity to witness the complete comn providing OSI products in the near f interoperability demonstrations were sh Lynch was invited to speak (about TCP/IP some impressions from the show.

In reponse to many requests I have com ments, etc. giving information about T *ConneXions* we will be reviewing some of t

The TCP/IP protocol suite uses 32-bit addr are divided into a network part and a l network part is called a *network number*. Center (NIC) at SRI International can using the TCP/IP protocol suite a unique request a network number assignme HOSTMASTER@SRI-NIC.ARPA or call 800

* In all fairness to our International subscribers, I should say "US standard binder," one area of standardization which the US has *not* actively participated in is that relating to paper sizes, A4 is close to but still different from "8.5x11").

CONNEXIONS
480 San Antonio Road
Suite 100
Mountain View, CA 94040
415-941-3399
FAX: 415-949-1779

FIRST CLASS MAIL
U.S. POSTAGE
PAID
SAN JOSE, CA
PERMIT NO. 1

CONNEXIONS

PUBLISHER	Daniel C. Lynch
EDITOR	Ole J. Jacobsen
EDITORIAL ADVISORY BOARD	Dr. Vinton G. Cerf, Vice President, National Research Initiatives.
	Dr. David D. Clark, The Internet Architect, Massachusetts Institute of Technology.
	Dr. David L. Mills, NSFnet Technical Advisor; Professor, University of Delaware.
	Dr. Jonathan B. Postel, Assistant Internet Architect, Internet Activities Board; Division Director, University of Southern California Information Sciences Institute.

CONNEXIONS

Subscribe to CONNEXIONS

U.S./Canada	$100. for 12 issues/year	$180. for 24 issues/two years	$240. for 36 issues/three years
International	$ 50. additional per year	(Please apply to all of the above.)	

Name ____________________ Title ____________________

Company ____________________

Address ____________________

City ____________________ State __________ Zip __________

Country ____________________ Telephone (   ) __________

☐ Check enclosed (in U.S. dollars made payable to CONNEXIONS.)

☐ Charge my ☐ Visa ☐ Master Card Card # ____________________ Exp. Date __________

Signature ____________________

Please return this application with payment to: CONNEXIONS
480 San Antonio Road Suite 100
Mountain View, CA 94040
415-941-3399 FAX: 415-949-1779

Back issues available upon request $15./each
Volume discounts available upon request